# BUNDESREPUBLIK DEUTSCHLAND



06452
1 of 1

## Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

**Aktenzeichen:** 100 29 904.0

**Anmeldetag:** 17. Juni 2000

**Anmelder/Inhaber:** Fa. Alcatel, Paris/FR

**Bezeichnung:** Verfahren und Sprachansagemodul zur Ausgabe einer Sprachansage sowie Programm-Modul dafür

**IPC:** G 10 L 13/00



**Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.**

München, den 21. November 2000
**Deutsches Patent- und Markenamt**
**Der Präsident**
Im Auftrag

Waasmaier

A 9161

Zusammenfassung

# Verfahren und Sprachansagemodul zur Ausgabe einer Sprachansage sowie Programm-Modul dafür

Die Erfindung betrifft ein Verfahren zur Ausgabe einer Sprachansage, deren Inhalt vom Daten-Inhalt mindestens eines Datensatzes abhängt, bei dem mittels eines ersten Programmsegments aufgrund des Daten-Inhalts ein Ansagetyp ausgewählt wird, bei dem ein zweites Programmsegment zur programmgesteuerten Ausgabe bereitgestellt wird, das für den ausgewählten Ansagetyp geeignet ist, bei dem die für das bereitgestellte zweite Programmsegment erforderlichen Parameter ermittelt werden und bei dem die dem ausgewählten Ansagetyp unter Berücksichtigung der ermittelten Parameter entsprechende Sprachansage ausgegeben wird. Die Erfindung betrifft weiter ein Sprachansagemodul und eine Programm-Modul dafür.

(Figur 1)

Datum: 16. Juni 2000
ZPL/S - Bs

**111342**

[Good morning, Good afternoon, Good evening], the number you have dialled has changed. Please dial the area code <*Area Code*> followed by the new number <*B-party number*>. { *loop infinite* | <*half second pause*> I repeat, please dial the area code <*Area Code*> followed by the new number <*B-party number*> }

Beispiel 1

Your call costs <*Dollar*> (_Dollar/Dollars) (_and_) <*Cent*> (_Cent/ Cents).

Beispiel 2

Your call { *DollarUnequalZero* | costs <*Dollar*> (Dollar/Dollars) | is free}

Beispiel 3

Your calls will be forwarded ; to your answer machine.

Beispiel 4

Your calls will be forwarded ; to your mobile telephone !

Beispiel 5

**Figur 1**

# Verfahren und Sprachansagemodul zur Ausgabe einer Sprachansage sowie Programm-Modul dafür

Die Erfindung betrifft ein Verfahren zur Ausgabe einer Sprachansage nach dem Oberbegriff des Anspruchs 1, ein Sprachansagemodul nach dem Oberbegriff des Anspruchs 2 und ein Programm-Modul nach dem Oberbegriff des Anspruchs 3.

Bekannte Sprachansagemodule sind konzipiert, um Ansagen mit einer einfachen Struktur auszugeben. Diese bestehen vorwiegend aus einem, möglicherweise geteilten, festen Teil und einem variablen Teil. Beispiele hierfür sind Zeitansagen ("Es ist jetzt", "fünf", "Uhr", "zweiunddreißig".), auch als Teil einer Fahrplanauskunft, die Ansage einer geänderten Rufnummer oder die Nennung der Kosten eines soeben beendeten Ferngesprächs.

Bei den bekannten Sprachansagemodulen sind Ansagen mit komplizierterer Struktur, etwa mit Alternativen, nicht automatisierbar möglich.

Der Erfindung liegt die Aufgabe zugrunde, auch Ansagen mit komplizierterer Struktur, etwa mit Alternativen, automatisierbar zu ermöglichen.

Diese Aufgabe wird erfindungsgemäß gelöst durch ein Verfahren nach der Lehre des Anspruchs 1, ein Sprachansagemodul nach der Lehre des Anspruchs 2 und ein Programm-Modul nach der Lehre des Anspruchs 3.

Der Erfindung liegt letztlich der Gedanke zugrunde, eine Programmiersprache zur Programmierung von Ansagen zu schaffen und diese dann zur Steuerung der Sprachansage zu verwenden.

Weitere Ausgestaltungen der Erfindung sind den Unteransprü-

chen und der nachfolgenden Beschreibung zu entnehmen.

Die vorgestellte Erfindung hat auch noch den Vorteil, daß die Programmsegmente nicht nur für die Steuerung von Ansagen verwendet werden können, sondern daß damit darüber hinaus auch die Dokumentation eines Sprachansagemoduls direkt möglich ist.

Im folgenden wird die Erfindung unter Zuhilfenahme der beiliegenden Zeichnungen weiter erläutert:

Figur 1 zeigt eine Reihe von Beispielen von Ansagetypen, aus denen durch das erfindungsgemäße Verfahren in einem erfindungsgemäßen Sprachansagemodul ausgewählt und die entsprechend den aktuellen Parametern variiert und ausgegeben werden.

Figur 2 zeigt eine weitere Reihe von Beispielen wie in Figur 1.

Figur 3 zeigt mehr symbolisch ein Ausführungsbeispiel eines PCs mit einem erfindungsgemäßen Sprachansagemodul.

Anhand von Beispiel 1 der Figur 1 wird zunächst kurz aufgezeigt, wie sich die Erfindung äußern kann.

Vorab soll aber zunächst ein Wort zur Sprache gesagt werden. (Dies ist gleichzeitig ein Hinweis an diejenigen Übersetzer, die später von der englischen Version ausgehend in weitere Sprachen übersetzen müssen.) Einerseits betrifft die Erfindung eine Sprachansage und beschäftigt sich schon deshalb mit Sprache und andererseits äußert sich die Erfindung in der Form einer Programmiersprache. Hier tritt die Sprache also nicht nur als Mittel zur Beschreibung der Erfindung, sondern auch als Inhalt und Ausdrucksform der Erfindung in Erscheinung.

Hier, im Beispiel 1 der Figur 1, wie auch in den meisten anderen Beispielen, wird unterstellt, daß die Sprachausgabe in englischer Sprache erfolgen soll. Die auszugebenden Sprachelemente, z. B. "Good morning, Good afternoon, Good evening" sind deshalb, als Ausdrucksform der Erfindung, in englisch wiedergegeben, obwohl die Beschreibung der Erfindung hier in deutscher

Sprache erfolgt. Das Gleiche gilt erst recht für den Programmcode als solchen, beispielsweise die Variablennamen, wie "<Area Code>", oder Anweisungen, wie "loop infinite". Es gilt aber auch für Anmerkungen, wie "where ThreeToneSuite is defined like" im Beispiel 8 in Figur 2, weil erfindungsgemäß auch nichtausführbare "Anweisungen" als Kommentare zulässig sind und diese dann in der Regel eben in englisch geschrieben werden.

Eine Ausnahme ist anhand der Beispiele 9 und 10 in Figur 2 dargestellt und wird im Zusammenhang mit der Beschreibung dieser Beispiele besprochen.

Was also in der ursprünglichen deutschsprachigen Version in deutscher Sprache zum Ausdruck gebracht wird, ist Beschreibung der Erfindung, während das, was in englischer Sprache zum Ausdruck gebracht wird, Inhalt oder Ausdrucksform der Erfindung ist. (In den weiteren Übersetzungen kann dies dann wieder ebenso unterschieden werden.) Aber auch in der englischsprachigen Version wird zumindest das, was in der Zeichnung in den Beispielen genannt und in der Beschreibung aus den Beispielen der Zeichnung zitiert wird, als Inhalt oder Ausdrucksform der Erfindung klar zu erkennen sein.

Selbstverständlich sind damit, wie bei Erfindungen in anderen Bereichen der Technik, länderspezifische Anpassungen nicht ausgeschlossen. Hier jedoch würden sich solche länderspezifischen Anpassungen in erster Linie durch das Ersetzen der Syntaxelemente durch solche äußern, die auf der jeweiligen Landessprache basieren.

Doch nun zurück zu Beispiel 1 der Figur 1:

Zunächst wird mit "[Good morning, Good afternoon, Good evening]" eine Anrede entsprechend der Tageszeit ausgewählt und ausgegeben, wobei die Auswahl automatisch erfolgt. Anschließend wird eine neue Rufnummer einschließlich der erforderlichen Vorwahl angesagt, indem zwei feste Satzbestandteile, "the number you have dialled has changed. Please dial the area code" und "followed by the new number", mit zwei Variablen, "<Area Code>" und "<B-party number>", abwechseln. In einer Endlosschleife,

"{loop infinite | <half second pause> I repeat, please dial the area code <Area Code> followed by the new number <B-party number> }", wird daraufhin solange die Ausgabe unter Hinzufügung einer Pause, "<half second pause>", und des Hinweises auf die Wiederholung, "I repeat", wiederholt, bis der Adressat der Ausgabe diese abbricht. Die Pause wird hier der Einfachheit halber als eine Variable mit vorgebbarer Länge, jedoch ohne Sprachinhalt ausgegeben.

In diesem Beispiel werden die Syntaxelemente "Anrede", "feste Satzbestandteile", "Variablen" und "Schleifen" benutzt und beispielhaft vorgestellt. Bevor auf weitere Beispiele eingegangen wird, wird kurz die Syntax dieser Programmiersprache zusammenfassend, jedoch nicht unbedingt vollständig, dargestellt; eine Ergänzung, Abwandlung oder auch eine Reduzierung soll nicht ausgeschlossen sein.

Beim Vergleich mit einer üblichen Programmiersprache wie C, C++ oder Pascal zeigt sich, daß die Programmiersprache, in der sich das erfindungsgemäße Verfahren äußert, ohne ausdrückliche Befehle ("explicit commands") auskommt. Sie besteht vielmehr aus Verweisen ("references") und Trennungen ("separators"). Die Verweise stellen dabei die Verbindungen (links) dorthin dar, wo sich die erforderliche Information befindet, während die Trennungen den Kontext definieren, in den die Verweise gestellt sind. Die Verweise, etwa eine Textfolge, sind nur im Zusammenhang mit einem Kontext von Bedeutung. Durch Verwendung einer bestimmten Trennung, etwa "<" oder ">", kann der Kontext und damit die Bedeutung des Verweises geändert werden.

In der Praxis wird ein automatischer Übersetzer die in natürlicher Sprache vorliegenden Verweise unter Zuhilfenahme einer besonderen Tabelle je nach vorliegendem Kontext in ein anwendungsspezifisches Format übertragen, das dann vom erfindungsgemäßen Sprachansagemodul gelesen werden kann. Um die Eingabe flexibler und lesbarer zu gestalten, unterstützt diese Programmiersprache auch Zeichensetzungssymbole ("punctuation symbols"). Dies sind unveränderliche Zeichen, die bei der Übersetzung keine Bedeutung haben. Sie können vom Benutzer hinzugefügt

oder entfernt werden, um das Aussehen und die Lesbarkeit zu verbessern, haben aber keine technische Auswirkung.

Auf diese Weise kann sehr einfach und auch nach nur kurzer Einarbeitung ein Sprachansagemodul programmiert werden, wobei das Ergebnis, also das Programm, sowohl unmittelbar für die Dokumentation geeignet ist als auch nach automatischer Übersetzung in eine niedrigere anwendungsspezifische Sprache übersetzt werden kann.

Eine Ansage ist eine Sammlung von festen Elementen ("elements"), wie "kein Anschluß unter dieser Nummer" und komplexen Segmenten ("Complex_Segments"), für die bei der konkreten Anwendung noch laufzeitspezifische Angaben, wie Parameter, erforderlich sind. Beispielsweise steht bei der Ansage "Der Anruf kostet <x> DM" das "<x>" für einen Parameter, der erst bekannt ist, wenn die Ansage wirklich erfolgt. Dabei ist "Laufzeit", (engl.: runtime) ein Ausdruck für denjenigen Augenblick, zu dem die Zusammenstellung zu einer aktuell auszugebenden Sprachansage tatsächlich erfolgt.

Im folgenden bedeutet (...) eine zusätzliche Möglichkeit ("option"), [...v...v...] eine Auswahl ("choice") und {...} eine Erweiterung ("extension").

Damit weist eine Ansage den folgenden Aufbau auf:

Ansage = (Prefix), Segment_List, (Suffix),

wobei:

Prefix, Suffix = [Element_List]

Segment_List = Segment {, Segment}

Element_List = Element {, ";", Element}

Parameter_List = Parameter {, ";", Parameter}, wobei ein Parameter ein Element, ein Segment, eine Zahl oder eine Variable, beispielsweise eine Laufzeitvariable, sein kann.

Element = [Textbeschreibung v "beliebiger_Text"]

Segment = [Element_List v Complex_Element v

Parameter_Fixed v Block_Segment] (, Punctuation sign) (, Flush sign "\")

Complex_Element = "<", Complex_Reference, ">"

Block_Segment = "{", Block_Reference, "|", Segment_List ({, "|", Segment_List}), "}"

IfThenElse_Segment = "{", Block_Reference oder Expression, "|", Segment_List_If (, "|", Segment_List_Else), "}"

Branch_Segment = "{", Block_Reference oder Expression, "|", Number_List, ":", Segment_List ({, "|", Number_List, ":", Segment_List}), (, "|", "ELSE :", Segment_List), "}". Zur Laufzeit wird "Block_Reference" oder "Expression" ermittelt und ein Sprung zu demjenigen Zweig durchgeführt, der die in der "Number_List" definierte Nummer trägt. Andernfalls erfolgt ein Sprung in den unter ELSE definierten Zweig, sofern ein solcher vorhanden ist.

Parameter_Fixed = "(", ("_"), Element_Singular (, ["/" v "\"], Element_Plural1 (, ["/" v "\"], Element_Plural2)), ("_"), ")". Das "_"-Zeichen steht für die Unterdrückung des vorausgehenden und/oder nachfolgenden Parameters, während das "/"-Zeichen für eine Übereinstimmung mit dem vorausgehenden oder nachfolgenden Parameter steht. Bei slavischen Sprachen muß ein zweiter Plural, Plural2, definiert werden.

Macro = "<<", Macro_Reference ({, "|", Parameter_List}), ">>"

Macro_Parameter = "[", Parameter_Number, "]". Steht für jede Einzelheit der Macro-Definition selbst.

Ein Ausdruck, "Expression", ist ein algebraischer Ausdruck, der zur Laufzeit ausgewertet wird und bekannte Operatoren aufweist, wie beispielsweise +, -, /, *, % (modulo), ! (not), and, or, xor, =, <>, <=, >=, <, > oder irgendwelche anderen systemspezifischen Operatoren. Beispielsweise ist "(Month=2) and (day>15)" ein gültiger Ausdruck.

Eine "Block_Reference" ist ein Verweis auf eine Einzelheit wie

1. eine Schleifenanweisung wie "loop 5 times" oder "loop infinite" (nur bei Ausgabe von Wörtern): ({loop|...}),

2. eine Anweisung für eine Dauer wie "10seconds" (nur bei Ausgabe von Wörtern): ({duration|...}),

3. eine Anweisung zur Wiederholung wie "loop as long the parameter buffer is not empty": ({repetitive|...}) oder

4. ein Ausdruck, der zur Laufzeit auszuwerten ist, wie "Euro=0 and Cent<>0", der zusammen mit dem IfThenElse_Segment ({Euro=0 and Cent<>0|(then)|(else)}) oder dem Branch_Segment ({Key pressed|1,2:...|5:...|ELSE:...}) benutzt wird.

Eine Complex_Reference ist ein Verweis auf eine komplexe Einzelheit wie

1. ein Parameterwert und die dazugehörige Regel, nach der er erzeugt wird,

2. eine (große) "Element_List" oder

3. jede komplexe Einzelheit, die als Merkmal vorliegt.

Eine "Macro_Reference" ist ein Verweis auf ein Makro, welches wiederum eine Sammlung von Elementen und Segmenten ist, sowie jede Ansage, bei der ein Element oder ein Teil davon symbolisch durch einen Parameter ersetzt werden kann. Ein Makro wird bei der Kompilation aufgelöst, wobei jeder Macro_Parameter durch seinen aus der Parameterliste "Parameter_List" entnommenen Wert ersetzt wird. Ein Makro arbeitet hier genauso wie in jeder bekannten höheren Programmiersprache, beispielsweise C++.

Eine "Parameter_Number" ist eine natürliche Zahl, die den n-ten Parameter der "Parameter_List" repräsentiert.

Ein "Punctuation sign" ist jedes Zeichen, das nicht dem Zeichensatz "/;<>{}()[]|" angehört. Es kann separat gespeichert werden und muß nicht Teil eines Elements sein (ein Element kann Bestandteil verschiedener Ansagen sein). Dieses Zeichen ist für die gesprochene Ansage ohne Bedeutung.

Das "Flush sign" soll wie ein Zeichen für Wagenrücklauf oder für eine neue Zeile behandelt werden. Es ist nur von Bedeutung,

wenn, beispielsweise zusätzlich zur Sprachausgabe, eine Textausgabe erfolgt. Beispielsweise kann als "Flush sign" der sogenannte "Backslash", "\", verwendet werden, ein Zeichen, das für andere Zwecke nicht gebraucht wird.

Bevor auf die technische Implementierung eingegangen wird, soll noch kurz auf die in den Figuren 1 und 2 gezeigten weiteren Beispiele eingegangen werden:

Beispiel 2 nach Figur 1 zeigt die Programmierung für die Ansage eines Preises, wobei unterstellt wird, daß der Anruf nicht völlig umsonst ist oder war. Dabei wird die Syntax des "Parameter_Fixed" verwendet. Zunächst erfolgt ein fester Ansagebestandteil, "Your call costs", gefolgt von einer Angabe in Dollar und Cent. Die aktuellen Werte hierfür, <Dollar> und <Cent>, werden dann eingesetzt, wenn diese Ansage abgerufen wird. Die zugehörigen Einheiten, Dollar, Dollars, Cent oder Cents, werden jeweils anschließend hinzugefügt, wobei noch je nach Wert die Singularform, Dollar oder Cent, oder die Pluralform, Dollars oder Cents, der Einheit ausgegeben wird. Ist einer der beiden Werte Null, so wird dieser nicht ausgegeben und auch die zugehörige Einheit unterdrückt. Das verbindende "and" wird sowohl unterdrückt, wenn der vorausgehende Wert, <Dollar>, Null ist als auch, wenn der nachfolgende Wert, <Cent>, Null ist.

Beispiel 3 in Figur 1 zeigt eine Ansage ähnlich der im Beispiel 2, jedoch nur auf ganze Dollar auf- oder abgerundet. Hier ist ein einfaches Unterdrücken von Angaben nicht mehr ausreichend. Deshalb werden hier durch das Branch_Segment DollarUnequalZero die Fälle unterschieden, daß etwas zu zahlen ist, "costs <Dollar> (Dollar/Dollars)", und daß insgesamt nichts zu zahlen ist, "is free". Die Unterdrückung der Einheit ist hier nicht erforderlich, weil dieser Zweig ohnehin nur angesprungen wird, wenn der Wert ungleich Null ist.

Für den Fall, daß einerseits genau abgerechnet werden soll, andererseits aber auch kostenlose Gespräche möglich sind, kann selbstverständlich eine Kombination beider Beispiele erfolgen,

wobei dann aber das Branch_Segment DollarUnequalZero anders definiert oder durch ein anderes Branch_Segment, etwa PriceUnequalZero ersetzt werden muß. Die Anweisungen DollarUnequalZero und PriceUnequalZero müssen dabei, wie bei Programmiersprachen üblich, an anderer Stelle, etwa als Funktion oder Unterprogramm definiert werden. Anstelle des Branch_Segments DollarUnequalZero könnte hier auch direkt die dahinter stehende Gleichung (Dollar • Zero) eingegeben werden. In solch einfachen Fällen mag es sich nicht lohnen, eine separate Funktion bereitzustellen und diese jeweils aufzurufen. Inwieweit dies auch für ein abgewandeltes Branch_Segment, etwa PriceUnequalZero gelten würde, hinge von dem hier nicht näher definierten inneren Aufbau ab.

Die folgenden drei Beispiele, Beispiel 4 und Beispiel 5 aus Figur 1 und Beispiel 6 aus Figur 2, zeigen Beispiele für mögliche Bestätigungen auf vom Benutzer vorgenommene Änderungen seines Benutzerprofils. Diese drei Beispiele zeigen jeweils einen ersten Ansageteil, in dem die Art der vorgenommenen Änderung bestätigt wird, "Your calls will be forwarded" oder "You entered a filter for the following directory numbers", und einen zweiten Teil, der den Inhalt der vorgenommenen Änderung bestätigt, nämlich im Falle der Rufum- oder -weiterleitung das jeweilige Ziel, "to your mobile telephone!" oder "to your answer machine." und im Falle der Eingabe einer Reihe von Rufnummern für ein Rufnummernfilter werden durch die Block_Reference {Repetitive|<DN>} alle eingegebenen Rufnummern zur Kontrolle ausgegeben.

Anhand des Beispiels 7 in Figur 2 soll eine komplexer aufgebaute Ansage dargestellt werden. Gezeigt wird eine Aussage für einen Wartenden in einer Warteschlange. Zunächst erfolgt eine Begrüßung: "Hello, welcome by XYZ. All lines are busy. Your call is being queued.", auf die hin wiederholt durch eine Block_Reference, {loop infinite |...}, auf die Tatsache hingewiesen wird, daß man sich gerade in einer Warteschlage befindet: "You are the <xth> in the queue". Dabei wird noch, als Variable "<xth>" angegeben, der wievielte man in dieser Warteschlange ist. Aus der Geschwindigkeit, mit der hier zurückge-

zählt wird, kann man abschätzen, wie lange man noch etwa warten muß. Um das Herannahen des Endes der Warteschlange deutlicher zu machen, ist durch eine weitere Block_Reference, {pool | 1: <music1> | 2: <music2> | ELSE: <wait tone>}, für eine Abwandlung der Aussage gegen Schluß gesorgt. Bei jedem Durchlauf der unendlichen Schleife "loop infinite" wird noch ein akustisches Zeichen ausgegeben, das aus einem Pool, "pool", ausgewählt wird und das im Normalfall, "ELSE", ein Wartezeichen, "<wait tone>", ist und das gegen Ende der Warteschlange durch zwei verschiedene Musikfolgen, erst "<music2>", dann "<music1>" ersetzt wird. Das Wartezeichen könnte dabei, wie im nachfolgenden Beispiel, aus einer vorab definierten Tonfolge bestehen.

Das Beispiel 8 in Figur 2 zeigt eine einfache Ausgabe für den Fall, daß unter der gewählten Rufnummer kein Anschluß besteht. Die Ausgabe wird zunächst durch die Ausführung eines Makrobefehls "<<ThreeToneSuite>>" angeführt und dann durch eine einfache Ausgabe "No abonnee on this number" abgeschlossen. In diesem Beispiel ist noch eine Möglichkeit angedeutet, Kommentare zu verwenden. Wie in jeder Programmiersprache kann auch hier vorgesehen werden, daß Kommentare eingefügt werden. Eine in vielen Programmiersprachen vorgesehene Möglichkeit, besteht darin, daß alles unbeachtet bleibt, was nach einem Anführungszeichen, "'", folgt; hier wird auf diese Weise der Kommentar 'where ThreeToneSuite is defined like "<Tone800Hz> ; <Tone1000Hz> ; <Tone1200Hz> ; <Pause500ms>" ' angefügt, um kurz den Inhalt dieses Makros zu erklären. Das Makro selbst muß selbstverständlich an anderer Stelle definiert werden.

Die letzten beiden Beispiele, Beispiel 9 und Beispiel 10 aus Figur 2, zeigen im Ergebnis dieselbe Ausgabe, nämlich die Angabe eines Kostenbetrags in einer slawischen Sprache, nämlich in Russisch. Slawische Sprachen haben zwei verschiedene Pluralbildungen, die bei der Ausgabe berücksichtigt werden müssen. (Genaugenommen wird hier der Genitiv Singular als zweite Pluralbildung verwendet.)

Um dies hier in einem nicht in einer slawischen Sprache vorliegenden Text zu zeigen, werden die Währungsangaben,

"Roubel/Roublei/Roublia", um die es hier geht, in russischer Sprache, aber ins Englische transliteriert, wiedergegeben. Der Rest, sowohl die Ansagen selbst, "Your call costs", wie auch Kommentare, ((Russian announcement)), und andere Programmelemente, <Costs>, werden in englischer Sprache dargestellt. Hier wird die Verwendung doppelter runder Klammern als weitere Art betrachtet, einen Kommentar einzufügen. Ein solcher Kommentar soll einfach überlesen werden, während bei einem Anführungszeichen "'" als Zeichen für einen Kommentar alles folgende nicht mehr beachtet werden soll.

Im Beispiel 10 in Figur 2 sind beide Arten von Kommentaren zu sehen. Im Beispiel 9 wird hier unterstellt, daß mit dem Aufruf der Variablen <Costs> automatisch das Makro für die Auswahl der richtigen Währungseinheit mit aufgerufen wird, während dies im Beispiel 10 explizit durch den Makroaufruf <<SlavicParFixed | <Costs> ; Roubel ; Roublei ; Roublia>> erfolgt. In diesem Beispiel 10 sind auch ausführlichere Kommentare hinzugefügt, beispielsweise ist die Definition des Makros als Kommentar beigefügt.

Nur wenn der Wert der Variablen <Costs> gleich "1" ist, wird als Währungseinheit die Singularform "Roubel" ausgegeben. Endet, bei einem Wert ungleich "1", der Wert der Variablen <Costs> auf 1, 2, 3 oder 4, nicht jedoch auf 11, 12, 13 oder 14, so wird die Form "Roublia" verwendet, während in allen anderen Fällen die Form "Roublei" verwendet wird.

Die verschiedenen Ansagen werden natürlich nicht so abgespeichert, wie in den Figuren 1 und 2 dargestellt. Insbesondere dient die hier verwendete graphische Darstellung, unter anderem auch mit verschiedenen Schriften, zwar dem besseren Verständnis, kann aber so nicht abgespeichert werden. Eine Sichtbarmachung in genau dieser Form ist aber mit einem entsprechend ausgebildeten Editor möglich. Die tatsächliche Speicherung erfolgt aber in Form von Tabellen.

So ist beispielsweise eine erste Tabelle für das Abspeichern von Ansagen als solchen vorgesehen. Diese Tabelle weist eine

erste Spalte mit eindeutigen Kennungen, eine zweite Spalte mit den Bezeichnungen der jeweiligen Ansagen, eine dritte Spalte mit der jeweils verwendeten Sprache, letztlich ein Kommentar, und eine vierte Spalte mit den Definitionen der Ansagen auf. Weiter kann eine Spalte mit Kommentaren vorgesehen sein. Stattdessen kann aber auch eine weitere Tabelle vorgesehen sein, die nur Kommentare enthält und über die eindeutigen Kennungen mit Zeilen dieser Tabelle oder auch mit Zeilen anderer Tabellen verknüpft ist.

Eine zweite Tabelle enthält die "Block_Segments", die Spalten für eindeutige Kennungen, für die Bezeichnungen, die "Block_References", und für die Definitionen dieser "Block_Segments" aufweist. Weitere Definitionen sind in weiteren Tabellen abgespeichert.

Abschließend soll noch kurz auf die mehr äußere Form der Realisierung eingegangen werden. Fig. 3 zeigt hierzu einen an sich bekannten PC, der durch geeignete Ausstattung als erfindungsgemäßes Sprachansagemodul mit einem erfindungsgemäßen Programm-Modul ausgestattet ist. Der in diesem Beispiel gezeigte Bildschirm als Ausgabemittel und die hier gezeigte Tastatur als Eingabemittel sind in der Grundausstattung nicht zwingend erforderlich. Das Sprachansagemodul ist dann Bestandteil eines sogenannten Servers, also eines Rechners, der nur im Verbund mit anderen Rechnern arbeitet und für diese bestimmte Aufgaben übernimmt.

Beispielsweise kann es ein Gebührenrechner des Betreibers eines Telekommunikationsnetzes sein, an den laufend Gebühreninformationen gesendet werden und der nicht nur für die monatlichen Gebührenabrechnungen verwendet werden soll, sondern nach Beendigung eines Gesprächs jeweils die angefallenen Kosten als Sprache ausgibt. Diese Kosten können aus dem Daten-Inhalt eines die Verbindungsdaten dieses Gesprächs enthaltenden Datensatzes ermittelt werden. Möglicherweise wird dann noch aus einem zweiten Datensatz, der spezifische Angaben über den Anschluß enthält, der diese Kosten verursacht hat, abgeleitet, in welcher Sprache die Ausgabe erfolgen soll. Beispiele hierfür sind die

Beispiele 2, 3, 9 und 10 aus den Figuren 1 und 2.

Dieser "Server" kann aber auch ein Vermittlungsrechner sein, der in einem sogenannten Call Center die ankommenden Anrufe auf die einzelnen Bedienplätze verteilt. Dabei muß ein Zugriff von Seiten des Sprachansagemoduls auf die Datensätze der in der Warteschleife wartenden Anrufe möglich sein, um für jeden einzelnen davon die aktuelle Position ermitteln und, wie im Beispiel 7 in Figur 2 gezeigt, als Sprache ausgeben zu können.

In beiden Fällen, wie auch in jedem anderen Fall, sind außer den angedeuteten Datenzugriffsmitteln noch Sprachausgabemittel erforderlich, mit denen die durch das Sprachansagemodul gebildeten Sprachansagen auch wirklich ausgegeben werden können. In irgendeiner an sich bekannten Art und Weise muß die Umsetzung der im Einzelfall zusammengestellten Sprachansage in gesprochene Sprache und deren Ausgabe über einen geeigneten (analogen oder digitalen) Anschluß an ein Telekommunikationsnetz erfolgen.

Der Datensatz, der die oben bereits erwähnten spezifischen Angaben über einen Anschluß enthält, wird oft auch als "Benutzerprofil" bezeichnet. Es kann nun zugelassen sein, daß ein Benutzer zumindest Teile dieses Datensatzes durch irgendeine Art der Ferneingabe selbst beeinflußt, und daß das Ergebnis dieser Ferneingabe anschließend als Sprache zur Kontrolle und Bestätigung wieder ausgegeben wird. Beispiele hierzu sind die Beispiele 4, 5 und 6 der Figuren 1 und 2.

Die Beispiele 1 und 8 der Figuren 1 und 2 zeigen zwei mögliche Ansagen eines Vermittlungsrechners, für den Fall, daß unter der gewählten Rufnummer kein Teilnehmer über diesen Vermittlungsrechner erreicht werden kann. Hier wird dann entweder, sofern bekannt, eine neue Teilnehmerrufnummer (Beispiel 1) ausgegeben oder es wird angesagt, daß unter dieser Teilnehmerrufnummer kein Eintrag vorliegt (Beispiel 8). Die Anfrage in der Teilnehmerdatenbank dieses Vermittlungsrechners enthält dann entweder einen Datensatz, der unter der bisherigen, jetzt gewählten Rufnummer einen Verweis auf die neue Rufnummer enthält

oder es kommt als Antwort auf den Datenzugriff des Datenzugriffsmittels als Sonderform eines Datensatzes eine Fehlermeldung zurück, die besagt, daß kein Eintrag vorliegt.

Häufig werden solche Rechner, die mit erfindungsgemäßen Sprachansagemodulen ausgerüstet sein können, aber auch gleichzeitig zur Pflege zumindest eines Teils der Daten verwendet, die Grundlage für die Sprachansagen sind. In all diesen Fällen, beispielsweise in Hotels oder Krankenhäusern, ändern sich die Daten laufend, wozu zwingend auch Ein- und Ausgabemittel, wie in Figur 3 auch dargestellt, erforderlich sind. Hier ändern sich auch häufig die organisatorischen Abläufe, was dann wiederum eine Anpassung der Sprachansagen erfordert. Gerade bei der vorliegenden Erfindung ist aber die Programmierung neuer oder die Abänderung bestehender Ansagen verhältnismäßig einfach und auch von wenig geschultem Personal durchführbar. Deshalb ist es für solche Fälle vorteilhaft, wenn das Sprachansagemodul auch mit geeigneten Editiermitteln ausgestattet ist. Hierfür können meist die in beinahe allen Rechnern ohnehin vorhandenen Editoren oder auch ein Schreibprogramm verwendet werden. In diesem Fall sollte der verwendete Rechner zusätzlich über Ein- und Ausgabemöglichkeiten (Mikrofon, Lautsprecher) für gesprochene Sprache verfügen, um Texte aufzusprechen und programmierte Ansagen zu testen.

Der Aufbau des erfindungsgemäßen Programm-Moduls zur Verwendung in einem erfindungsgemäßen Sprachansagemodul zur Durchführung des erfindungsgemäßen Verfahrens zur Ausgabe einer Sprachansage ist an sich üblich. Es liegt im Rahmen üblicher Programmiertätigkeit, ein Programm in Segmente (Subroutines, Procedures) aufzuteilen, wobei einzelne Segmente mehr für die Steuerung des Ablaufs und andere für die Durchführung einzelner Tätigkeiten bestimmt sind, dieses Programm mit Treibern zum Datenzugriff und zur Sprachausgabe zu verbinden und mit den oben dargestellten erfinderischen Inhalten zu füllen.

Patentansprüche

1. Verfahren zur Ausgabe einer Sprachansage, deren Inhalt vom Daten-Inhalt mindestens eines Datensatzes abhängt, **dadurch gekennzeichnet, daß** mittels eines ersten Programmsegments aufgrund des Daten-Inhalts ein Ansagetyp ausgewählt wird, daß ein zweites Programmsegment zur programmgesteuerten Ausgabe bereitgestellt wird, das für den ausgewählten Ansagetyp geeignet ist, daß die für das bereitgestellte zweite Programmsegment erforderlichen Parameter ermittelt werden, und daß die dem ausgewählten Ansagetyp unter Berücksichtigung der ermittelten Parameter entsprechende Sprachansage ausgegeben wird.

2. Sprachansagemodul zur Ausgabe einer Sprachansage, deren Inhalt vom Daten-Inhalt mindestens eines Datensatzes abhängt, **dadurch gekennzeichnet, daß** das Sprachansagemodul eine Programmsteuerung aufweist, daß ein erstes Programmsegment vorhanden ist, mittels dessen aufgrund des Daten-Inhalts ein Ansagetyp auswählbar ist, daß jedem Ansagetyp ein zweites Programmsegment aus einer Vielzahl zweiter Programmsegmente zugeordnet ist, daß Datenzugriffsmittel vorhanden sind, um die für ein ausgewähltes zweites Programmsegment erforderlichen Parameter aus dem mindestens einen Datensatz zu ermitteln, und daß Sprachausgabemittel vorhanden sind, um die dem ausgewählten Ansagetyp unter Berücksichtigung der ermittelten Parameter entsprechende Sprachansage auszugeben.

3. Programm-Modul zur Verwendung in einem Sprachansagemodul zur Ausgabe einer Sprachansage, deren Inhalt vom Daten-Inhalt mindestens eines Datensatzes abhängt, **dadurch gekennzeichnet, daß** das Programm-Modul ein Programmsteue-

rungssegment, ein erstes Programmsegment und eine Vielzahl zweiter Programmsegmente aufweist, daß das Programmsteuerungssegment zur Steuerung des Zugriffs auf Datenzugriffsmittel und Sprachausgabemittel sowie zur Steuerung des Aufrufs des ersten Programmsegments und eines Programmsegments aus der Vielzahl der zweiten Programmsegmente ausgestaltet ist, daß das erste Programmsegment derart ausgestaltet ist, daß damit ein Ansagetyp und ein diesem Ansagetyp zugeordnetes zweites Programmsegment aus der Vielzahl zweiter Programmsegmente auswählbar ist, und daß jedes der zweiten Programmsegmente aus der Vielzahl zweiter Programmsegmente derart ausgestaltet ist, daß es unter Zuhilfenahme der erforderlichen Parameter, die mittels des Programmsteuerungssegments und der Datenzugriffsmittel aus dem mindestens einen Datensatz ermittelt werden, eine aktuelle Sprachansage formt, und daß es mittels des Programmsteuerungssegments und der Sprachausgabemittel diese Sprachansage zur Ausgabe bringt.

4. Sprachansagemodul nach Anspruch 2, dadurch gekennzeichnet, daß Editiermittel vorhanden sind, mittels derer bestehende Ansagetypen mit zugehörigen zweiten Programmsegmenten verändert oder gelöscht oder neue Ansagetypen mit zugehörigen zweiten Programmsegmenten erzeugt werden können, und daß Ein- und Ausgabemittel vorhanden sind, über die mit den Editiermitteln kommuniziert werden kann.



Datum: 16. Juni 2000
ZPL/S - Bs

[Good morning, Good afternoon, Good evening], the number you have dialled has changed. Please dial the area code <*Area Code*> followed by the new number <*B-party number*>. { *loop infinite* | <*half second pause*> I repeat, please dial the area code <*Area Code*> followed by the new number <*B-party number*> }

Beispiel 1

Your call costs <*Dollar*> (_Dollar/Dollars) (_and_) <*Cent*> (_Cent/ Cents).

Beispiel 2

Your call { *DollarUnequalZero* | costs <*Dollar*> (Dollar/Dollars) | is free}

Beispiel 3

Your calls will be forwarded ; to your answer machine.

Beispiel 4

Your calls will be forwarded ; to your mobile telephone !

Beispiel 5

**Figur 1**

You entered a filter for the following directory numbers: { *Repetitive* | *<DN>* }

Beispiel 6

Hello, welcome by XYZ. All lines are busy. Your call is being queued. { *loop infinite* | You are the *<xth>* in the queue { *pool* | 1 : *<music1>* | *2 : <music2>* | ELSE : *<wait tone>* }}

Beispiel 7

*<<ThreeToneSuite>>* No abonnee on this number 'where ThreeToneSuite is defined like "*<Tone800Hz>* ; *<Tone1000Hz>* ; *<Tone1200Hz>* ; *<Pause500ms>*"

Beispiel 8

((Russian announcement)) Your call costs *<Costs>* (Roubel/Roublei/Roublia)

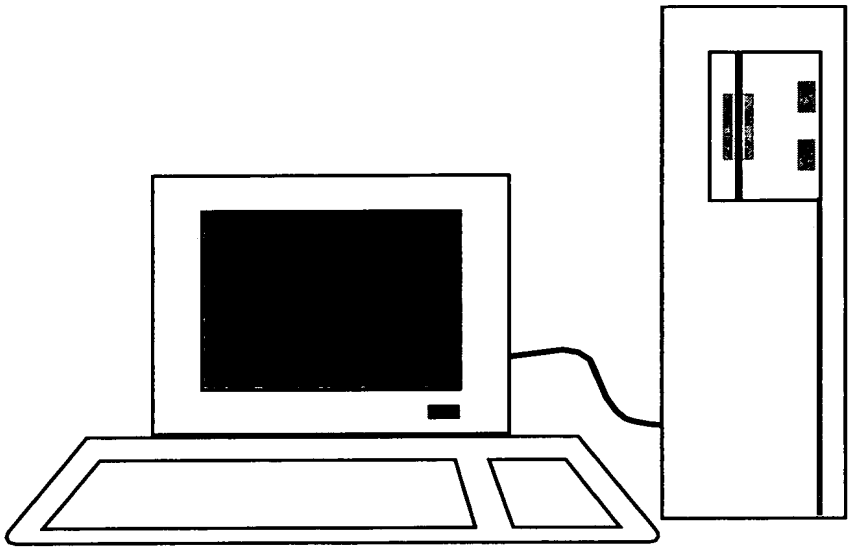
Beispiel 9

((Russian announcement. Using a simulation via macro of the Slavic Parameter_Fixed Segment in the example above)) Your call costs *<Costs>* *<<SlavicParFixed* | *<Costs>* ; Roubel ; Roublei ; Roublia>> 'using the macro template SlavicParFixed defined as {[1]| 1 : [2] | ELSE:{(([1]) % 100) | 11, 12, 13, 14 : [3] | ELSE : { (([1]) % 10) | 1, 2, 3, 4 : [4] | ELSE : [3] }}}

Beispiel 10

**Figur 2**

111342



**Figur 3**